

REMARKS

In the Final Office Action, the Examiner provisionally rejected claims 8 and 18 based on a judicially created doctrine of obviousness-type doubling patenting as being unpatentable over claims 4, 12, and 19 of copending U.S. Patent Application No. 10/676,374 of Cherdrone et al. ("Cherdrone"), and rejected claims 1-22 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,964,053 to Ho et al. ("Ho").

By this Amendment, Applicants have amended claims 1, 2, 10, 11, and 15-18. Claims 1-22 are currently pending. Based on the following remarks, Applicants respectfully traverse the Examiner's § 102(e) rejection.

A. Double Patenting Rejections

The Examiner provisionally rejected claims 8 and 18 under the judicially created doctrine of obviousness-type double patenting over claims 4, 12, and 19 of Cherdrone. Applicants respectfully request that the Examiner hold all double-patenting rejections in abeyance until the indication of otherwise allowable subject matter. Upon review of the remarks made in this paper, should the Examiner believe this application to be in condition for allowance but for the double patenting rejections held in abeyance, Applicants respectfully request that the Examiner contact the undersigned representative to discuss an appropriate resolution.

B. Rejections of Claims 1-22 Under 35 U.S.C. § 102(e)

1. Claims 1-9

Applicants respectfully traverse the rejection of claim 1 under 35 U.S.C. § 102(e) as being anticipated by Ho for at least the reason that Ho fails to disclose every claim

element recited in claim 1. Amended independent claim 1 recites, among other things, “generat[ing] an API using [a] set of intermediate objects as inputs, wherein the API enables accessing . . . development objects.” The Examiner interpreted “claim 1 as a scenario having 3 steps: (i) receive a first data model in a first language; (ii) generate **intermediate objects** based thereupon; (iii) generate API for **accessing these objects** based on those objects and some code template.” Office Action at 9. (emphasis added). Applicants respectfully submit that the Examiner misinterpreted claim 1 because the third step (iii) involves generating an API to **access the development objects**, and not the intermediate objects as interpreted by the Examiner. The rejection of claims 1-9 is based on the Examiner’s misinterpretation of claim 1.

Ho is directed to “[a] method of and a system for processing an enterprise application request on an end user application and an application server.” Ho, Abstract. “The end user application and the application server have at least one connector between them, and the steps of (i) converting the application request from the language of the end user application (as a source language), and (ii) converting the response to the application request from the language running on the application server (as a source language) to the language of the end user application (as a target language).” Id. This is achieved by “invoking connector metamodels of the respective source and target languages, populating the connector metamodels with metamodel data of each of the respective source and target languages, and converting the source language to the target language.” Id.

The Examiner asserted that the connector of Ho, which calls “application API . . . to access the objects defined as metamodel type code template inside

intermediate OTMA messages,” discloses the “generat[ing] an API” feature recited in amended independent claim 1. Office Action at 5. Applicants respectfully disagree.

Ho defines a connector as “a dynamic, **run-time**, interface between platforms that stores the functions and parameters of the target platform or program, and **binds with the target platform program in real time.**” Ho, col. 9, lines 25-29. (emphasis added). Thus, the connector of Ho has nothing to do with accessing **development objects** that represent building blocks for **developing** an application.

The Examiner also asserted that Fig. 7 discloses a connector API reading on creating and calling an API to access objects defined as metamodel type code template inside intermediate OTMA messages whose inputs/outputs are processed. Office Action at 5. Fig. 7 of Ho illustrates a run-time connector 701 connecting a middleware 713 and an existing application program 703. The connector 701 interfaces with the application program 703 through an interface 705. Ho, col. 11, lines 26-29. The interface 705 contains the interface definition of the application 703 in accordance with an application interface metamodel 707. Ho, col. 11, lines 29-31. “The interfaces 705 are described in terms of the Application Interface Metamodel.” Ho, col. 11, lines 38-39.

However, Fig. 7 and the description of the figure merely disclose invoking the interface 705. The reference fails to disclose generating an API or the interface 705. Also, nowhere does Ho disclose that the interface 705 is generated using a set of intermediate objects as inputs or that the interface 705 enables accessing development objects.

Furthermore, the Examiner's assertion is based on the Examiner's misinterpretation of claim 1, as explained above. Thus, even if the Examiner's assertion were true, the interface 705 is allegedly invoked to access merely **intermediate** OTMA messages, and not development objects. The Examiner failed to establish that the intermediate OTMA messages are equivalent to development objects representing building blocks for developing an application. Therefore, Fig. 7 and the description of the figure fail to disclose "generat[ing] an API using [a] set of intermediate objects as inputs, wherein the API enables accessing . . . development objects," as recited in amended independent claim 1.

The Examiner further asserted that Ho teaches "based on the intermediate messages[, retrieving] more definition or data structures therein . . . using connector being invoked or triggered as APIs to obtain meta-definition in the parsing/validating process." Office Action at 10. Again, the Examiner's assertion is based on the Examiner's misinterpretation of claim 1, as explained above. Thus, even if the Examiner's assertion were true, the connector is being allegedly invoked or triggered to obtain meta-definition in the parsing/validating process. As set forth above, the connector is a **run-time** interface and thus does not enable creating, modifying, and accessing **development objects**. Moreover, nowhere does the Office Action establish that meta-definition in the parsing/validating process is equivalent to development objects representing building blocks for developing an application.

For at least the reasons set forth above, Ho does not support the § 102 rejection of amended independent claim 1. In addition, because claims 2-9 depend from amended independent claim 1, claims 2-9 are allowable at least by virtue of their

dependence from an allowable claim. Accordingly, Applicants respectfully request reconsideration and withdrawal of the § 102(e) rejection of claims 1-9 based on Ho.

2. Claims 10-17

Applicants respectfully traverse the rejection of claim 10 under 35 U.S.C. § 102(e) as being anticipated by Ho for at least the reason that Ho fails to disclose the combination of steps recited in claim 10. Amended independent claim 10 recites, among other things, “generat[ing] a set of intermediate objects using the first model; and generat[ing] an XML schema using the set of intermediate objects as inputs, wherein the XML schema enables implementing the development objects.”

The Examiner asserted that Source 507 and/or XML instance file (613) disclose “generat[ing] a set of intermediate objects using the first model.” Although Source 507 is an input to Generation Tool 509 to generate XML Document 511 as illustrated by Fig. 5, nowhere does Ho disclose that Source 507 is generated using the first model (i.e., Rose documents and/or Rose File, as asserted by the Examiner). Further, even if Source 507 can be considered as a set of intermediate objects that are generated using the first model, Source 507 is an input to generate an XML document 511, which is not an XML Schema. An XML document and an XML schema serve different purposes, and are materially different. An XML schema defines rules or constraints for XML documents, and XML documents are constructed in compliance with the rules and constraints defined in an XML schema. Thus, Source 507 is neither an intermediate object being generated using the first model nor an intermediate object being used as an input to generate an XML schema.

Moreover, Fig. 6 depicts XMI instance file (613) as a final output and not as an intermediate output, and Fig. 8 merely depicts XMI instance file (613) being stored in XML repository 809. Thus, in both Figs. 6 and 8, XMI instance file (613) is not shown as an input to generate an XML schema. Accordingly, Ho fails to support the Examiner's assertion that XMI instance file (613) is an intermediate object being used as an input to generate an XML schema.

Ho also makes it clear in Fig. 6 that there is no intermediate object between the first model (i.e., CAM Model/Rose File 601, as asserted by the Examiner) and an XML schema (i.e., DTD and Schema for Rose Model 605, as asserted by the Examiner). CAM Model/Rose File 601 is an input to Toolkit 603, and Toolkit 603 generates DTD and Schema for Rose Model 605. Thus, Ho fails to disclose "generat[ing] a set of intermediate objects using the first model; and generat[ing] an XML schema using the set of intermediate objects as inputs, wherein the XML schema enables implementing the development objects," as recited in amended independent claim 10.

For at least the reasons set forth above, Ho does not support the § 102 rejection of amended independent claim 10. In addition, because claims 11-17 depend from amended independent claim 10, claims 11-17 are allowable at least by virtue of their dependence from an allowable claim. Accordingly, Applicants respectfully request reconsideration and withdrawal of the § 102(e) rejection of claims 10-17 based on Ho.

3. Claims 18-22

The Examiner rejected claim 18 for the same reasons that the Examiner rejected independent claim 1. As explained above, Ho does not support the § 102(e) rejection of amended independent claim 1. Furthermore, amended independent claim 18 recites an

additional feature, “us[ing an] API to perform operations on a development object representing a building block for developing [an] application.” The Examiner asserted that “[w]hen development objects are retrieved by means of connector via MQ API or database queries remote calls, such retrieval can be analogized as being performed on the stored data in a way that the data has to be marshaled in a message form to be sent back to the requesting calls.” Office Action at 12. As explained above, however, a connector is a **run-time** interface between two platforms that are **already developed**, and has nothing to do with performing operations on a **development object** representing a building block for **developing** an application. Thus, the operations that are performed by the connector, as allegedly by the Examiner, are not operations performed on a development object. For at least these additional reasons, Applicants respectfully request reconsideration and withdrawal of § 102(e) rejection of independent claims 18 based on Ho.

Claims 19-22 depend from amended independent claim 18. Therefore, claims 19-22 are allowable at least by virtue of their dependence from an allowable claim. Accordingly, Applicants respectfully request reconsideration and withdrawal of § 102(e) rejection of claims 19-22 based on Ho.

C. Conclusion

In view of the foregoing amendments and remarks, Applicants respectfully request reconsideration and reexamination of this application and the timely allowance of the pending claims.

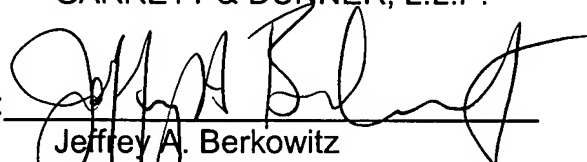
Please grant any extensions of time required to enter this response and charge any additional required fees to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: June 4, 2007

By:


Jeffrey A. Berkowitz
Reg. No. 36,743